ธานินทร์ ลีลาคุณารักษ์

SOFT MAIL

ไร่เกษตรธรรม ไทย กราฟิก ไลบรารี : สร้างตัวอักษร

นำเสนอแนวคิด และวิธีการสร้างตัวอักษร ที่ใช้ในระบบไร่เกษตรธรรม ไทย กราฟิก ไลบรารี

ผู้เขียนได้มีโอกาสพัฒนาโปรแกรมใช้งานที่เป็นภาษาไทยมาหลายโปรแกรม แต่เดิมจะใช้ไดรเวอร์ของบริษัทที่ผลิตการ์ดภาษาไทย บางครั้งเกิดปัญหาลูกค้ามีการ์ดไม่ตรงกับเรา ทำให้การทำงานของโปรแกรมผิดปกติไป หลังจากที่มีเวิร์ดจุฬาฯ ออกมา ซึ่งเป็นการจัดการภาษาไทยในกราฟิกโหมด ผู้เขียนได้ใช้แนวความคิดนี้ในการเขียนโปรแกรมแทนแบบเก่า แต่ไม่ได้รวบรวมเป็นชุดไว้โดยเฉพาะ เมื่อมีเวลาจึงได้ทำการรวบรวมเป็นชุดไลบรารีที่จัดการภาษาไทยในกราฟิกโหมดขึ้น สำหรับในตอนแรกนี้ผู้เขียนขอกล่าวถึงแนวคิดที่ใช้ในระบบ ไร่เกษตรธรรม ไทย กราฟิก ไลบรารี ก่อน

ที่มาของตัวอักษรแบบที่ใช้กันทั่วไปในกราฟิกโหมด

ลักษณะตัวอักษรที่ใช้ในเวิร์ดจุฬาฯ นั้นจะเก็บในรูป bitimage 1 ตัวอักษรจะมีขนาด 8 × 20 จุด (เฉพาะตัวอักษรที่แสดงบนจอภาพแบบปกติ) โดย 1 จุดจะใช้หน่วยความจำ 1 บิต ในการเก็บข้อมูลในระบบไมโครคอมพิวเตอร์ (IBM PC) ที่เราใช้อยู่ทั่วไป 1 ไบต์จะใช้เนื้อที่ 8 บิต ดังนั้น การเก็บตัวอักษรขนาด 8×20 จุด จะใช้ 20 ไบต์ พอดี ถ้าเราจะสร้างตัวอักษรลักษณะนี้ 1 ตัว จะมีขั้นตอนการทำดังนี้

1) สร้างตารางขนาด 8×20 แล้วใส่ค่า 1 ลงในตาราง โดยจะใส่ในช่องที่เราจะแสดงรูปตัวอักษร สมมติว่าจะสร้างตัว ท. ทหาร ก็จะได้ลักษณะดังตารางที่ 1

	1	2	3	4	5	6	7	8
1								
2								
3								
4								
5								
6								
7								
8								
9	1	1				1		
10	1		1		1		1	
11		1		1			1	
12		1	1				1	
13		1					1	
14		1					1	
15		1					1	
16		1					1	
17								
18								
19								
20								

ตารางที่ 1
แสดงการสร้างตัวอักษรตัว "ท"

ตารางสร้างตัวอักษรจะแบ่งเป็น 3 ส่วนดังนี้

- แถวที่ 1 ถึง 8 จะใช้สำหรับสระบนรวมถึงหางของตัวอักษรบางตัว
- แถวที่ 9 ถึง 16 ใช้สร้างตัวอักษร และสระทั่วไป
- แถวที่ 17 ถึง 20 ใช้สำหรับสระล่าง และฐานของตัวอักษรบางตัว

2) จากตารางข้างต้นเรามองในรูปแบบของเลขฐานสอง ซึ่งมีค่า 0 และ 1 โดยช่องว่างหมายถึงค่า 0 จากนั้นแปลงเป็นเลขฐาน 16 จะได้เป็นตัวเลขฐานสิบหก 2 หลัก จากตารางที่ 1 เราจะได้ดังนี้ดังตารางที่ 2

3. นำตัวเลขฐาน 16 ที่ได้มา เก็บไว้ในตัวแปรอะเรย์ จากข้างต้นจะได้ดังรูปที่ 1

ตัวอย่างที่ท่านเห็นนี้มีเพียงแค่ 1 ตัวอักษร ตัวอักษรที่ใช้จริงทั้งหมดมี 256 ตัวอักษร อะเรย์ที่ใช้จะมีขนาด 20×256

```
1 —> 0000 0000 = แปลงเป็น เลขฐาน 16 => 0×00
2 —> 0000 0000 = แปลงเป็น เลขฐาน 16 => 0×00
3 —> 0000 0000 = แปลงเป็น เลขฐาน 16 => 0×00
4 —> 0000 0000 = แปลงเป็น เลขฐาน 16 => 0×00
5 —> 0000 0000 = แปลงเป็น เลขฐาน 16 => 0×00
6 —> 0000 0000 = แปลงเป็น เลขฐาน 16 => 0×00
7 —> 0000 0000 = แปลงเป็น เลขฐาน 16 => 0×00
8 —> 0000 0000 = แปลงเป็น เลขฐาน 16 => 0×00
9 —> 1100 0100 = แปลงเป็น เลขฐาน 16 => 0×C4
10 —> 1010 1010 = แปลงเป็น เลขฐาน 16 => 0×AA
11 —> 0101 0010 = แปลงเป็น เลขฐาน 16 => 0×52
12 —> 0110 0010 = แปลงเป็น เลขฐาน 16 => 0×62
13 —> 0100 0010 = แปลงเป็น เลขฐาน 16 => 0×42
14 —> 0100 0010 = แปลงเป็น เลขฐาน 16 => 0×42
15 —> 0100 0010 = แปลงเป็น เลขฐาน 16 => 0×42
16 —> 0100 0010 = แปลงเป็น เลขฐาน 16 => 0×42
17 —> 0000 0000 = แปลงเป็น เลขฐาน 16 => 0×00
18 —> 0000 0000 = แปลงเป็น เลขฐาน 16 => 0×00
19 —> 0000 0000 = แปลงเป็น เลขฐาน 16 => 0×00
20 —> 0000 0000 = แปลงเป็น เลขฐาน 16 => 0×00
```

ตารางที่ 2
แสดงการแปลงเป็นตัวเลขฐาน 16 จากตารางที่ 1

```
char ch[20] = { 0×00, 0×00, 0×00, 0×00,
                0×00, 0×00, 0×00, 0×00,
                0×C4, 0×AA, 0×52, 0×62,
                0×42, 0×42, 0×42, 0×42,
                0×00, 0×00, 0×00, 0×00 };
```

รูปที่ 1 แสดงการเก็บเลขฐาน 16 มาเก็บไว้ในตัวแปรอะเรย์

การนำตัวอักษรข้างต้นมาแสดง

สำหรับการนำตัวอักษรมาแสดงนั้นมีวิธีทำหลายวิธี (ขอให้ดูจากบทความของ อ.อาจหาญ สัตยารักษ์ เรื่อง เทคนิคการสร้างอักขระกราฟิก วารสารไมโครคอมพิวเตอร์ ฉบับที่ 75 เดือนตุลาคม 2535 หน้า 291) ส่วนวิธีที่

ค่อนข้างจะนิยมใช้หรือใช้กันมานานแล้วคือ นำค่าในตารางตัวอักษรที่เราสร้างไว้ มาเปรียบเทียบชุดข้อมูลชุดหนึ่งเพื่อตรวจสอบทีละบิตแล้วเขียนบนจอภาพทีละจุด โดยใช้คำสั่ง putixel ดังตัวอย่างคำสั่งในลิสติงที่ 1

ในบรรทัดที่ 8 จะนำข้อมูลจาก ch 1 ไบต์มา and(&) กับ ข้อมูลใน ARRAY bitcheck ทั้ง 8 ค่า นั่นคือ ตรวจสอบบิตที่จะแสดง ถ้ามีค่าเป็น 1 ก็จะให้ทำ คำสั่ง putpixel ในกรณีที่เป็น 0 ก็จะข้ามไป ลักษณะที่คอมพิวเตอร์ทำงานจะมีลักษณะดังตัวอย่าง เช่น ch[0] & bitcheck[0] จะได้ดังรูปที่ 2

จากตัวอย่างโปรแกรมจะไม่ลงจุด จากนั้นจะเอา ch[0] and กับ bitcheck ตัวถัดไปจนครบ 8 ค่า แล้วถึงขยับไปที่ ch ตัวถัดไป

การสร้างตัวอักษรที่ใช้ในระบบไร่เกษตรธรรม ไทย กราฟิก ไลบรารี

ตัวอักษรที่ใช้ในระบบนี้ก็เป็นตัวอักษรที่ได้จากเวิร์ดจุฬาฯ เช่นกัน แต่ต่างกันตรงที่ระบบไร่เกษตรธรรม ไทย กราฟิก ไลบรารี จะมองตัวอักษรเป็นรูปภาพรูปหนึ่ง นั่นคือ เราจะนำฟอนต์ของเวิร์ดจุฬาฯ มาสร้างตัวอักษรชุดใหม่ โดยใช้คำสั่งในการเก็บที่มีในตระกูลเทอร์โบ คือ getimage และนำตัวอักษรมาแสดงโดยคำสั่ง putimage โปรแกรม makepat.c ในลิสติง 2 เป็นตัวอย่างโปรแกรมที่ใช้ในการสร้างตัวอักษรตามแนวคิดที่กล่าวมาข้างต้น

ลิสติงที่ 1 ตัวอย่างของคำสั่งที่ใช้แสดงตัวอักษรบนจอภาพ

```
showchar(int x,int y,unsigned char *ch)
{
   char bitcheck[8] = { 0x80,0x40,0x20,0x10,0x08,0x04,0x02,0x01 };
   int x1,y1;
   for (y1 = 0;y1 < 20;y1++) {
      for (x1 = 0;x1 < 8;x1++) {
         if (ch[y1] & bitcheck[x1]) != 0) {
            putpixel((x + x1),(y + y1),YELLOW);
         }
      }
   }
}  /*end*/
```

ch[0]	= 0 0 0 0 0 0 0 0
(&)	& & & & & & & &
bitcheck	= 1 0 0 0 0 0 0 0
ผลที่ได้	= 0 0 0 0 0 0 0 0

รูปที่ 2

โปรแกรมในลิสติงที่ 2 เป็นโปรแกรมที่ใช้ในการสร้างตัวอักษร ลักษณะรูปแบบการเขียนโปรแกรมไม่ดีนัก ทั้งนี้เพื่อให้เห็นภาพของการทำงานของการสร้างตัวอักษรรูปภาพชัดเจนขึ้น โดยจะแสดงแถบแสงขณะเก็บภาพตัวอักษรหนึ่ง ๆ ด้วย รายละเอียดโปรแกรมมีดังนี้

- ฟังก์ชัน loadfont (บรรทัดที่ 12 ถึง 23) ทำหน้าที่ โหลดฟอนต์ของเวิร์ดจุฬาฯ มาไว้ที่ตัวแปรอะเรย์ชื่อ font

- ฟังก์ชัน PutBitMap (บรรทัดที่ 25 ถึง 36) ทำหน้าที่ในการนำตัวอักษรจากฟอนต์มาแสดงบนจอภาพ การทำงานจะเหมือนกับคำสั่ง showchar( ) ในลิสติงที่ 1

- ฟังก์ชัน savepattern (บรรทัดที่ 38 ถึง 48) ทำหน้าที่เก็บภาพของตัวอักษรลงแฟ้มข้อมูลทีละตัว สำหรับบรรทัดที่ 43, 45 และ 46 เป็นคำสั่งที่เพิ่มขึ้นมาเพื่อให้เห็นการทำงานของโปรแกรมเท่านั้น โดยแสดงแถบแสงขณะเก็บภาพตามที่กล่าวมาข้างต้น และถ่วงเวลาเล็กน้อยเพื่อไม่ให้โปรแกรมทำงานเร็วเกินไป

- ฟังก์ชัน opengraph (บรรทัดที่ 50 ถึง 55) ใช้เรียกระบบกราฟิกเขียนรวมไว้เพื่อให้ง่ายต่อการเรียกใช้

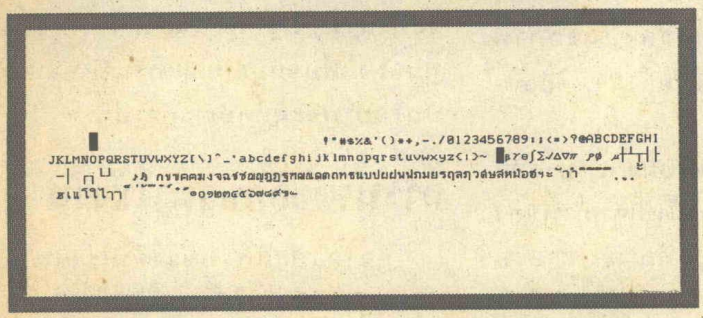
- ฟังก์ชัน main (บรรทัดที่ 57 ถึง 89) คำสั่งในบรรทัดที่ 64 จะสร้างแฟ้มข้อมูลชื่อ NORMAL.CHR เพื่อเก็บรูปตัวอักษร บรรทัดที่ 65 ถึง 73 จะนำตัวอักษรของเวิร์ดจุฬาฯทั้งหมดมาแสดงบนจอภาพ บรรทัดที่ 74 ถึง 84 จะทำการเก็บรูปภาพของตัวอักษรทั้งหมดลงแฟ้มข้อมูล ขณะที่รันโปรแกรมบนจอภาพจะมีลักษณะดังรูปที่ 3

การนำตัวอักษรมาใช้งาน

จากโปรแกรมข้างต้นเราจะได้ตัวอักษรซึ่งเก็บไว้ในแฟ้มข้อมูลชื่อ

ลิสติงที่ 2 โปรแกรม makepat.c

```
/* makepat.c */
#include <graphics.h>
#include <fcntl.h>
#include <alloc.h>
/***************************/
#define BUFSIZE       5120
/***************************/
unsigned char x,y;
char font[256][20];
unsigned char ptr;
int h2;
/***************************/

void loadfont()
{
  int h1;
  if ((h1 = open("normal.fon",O_RDONLY)) != -1) {
    if (read(h1,font,BUFSIZE) == -1) {
       exit(1);
    }
  }else{
    exit(1);
  }
  close(h1);
}

PutBitMap(int x,int y,int chn,int color)
{
  char BitMap[8] = {0x80,0x40,0x20,0x10,0x08,0x04,0x02,0x01};
  register int x1,y1;
  for (y1 = 0;y1 < 20;y1++) {
      for (x1 = 0; x1 < 8;x1++) {
          if ((font[chn][y1] & BitMap[x1]) != 0) {
             putpixel((x+x1),(y+y1),color);
          }
      }
  }
}

savepattern(int x,int y)
{
  void *bit;
  bit = malloc(imagesize(x,y,x+8,y+20));
  getimage(x,y,x+8,y+20,bit);
  putimage(x,y,bit,NOT_PUT);
  write(h2,bit,(imagesize(x,y,x+8,y+20)));
  delay(300);
  putimage(x,y,bit,COPY_PUT);
  free(bit);
}

void opengraph()
{
   int mode,driver;
   driver = DETECT;
   initgraph(&driver,&mode,"");
}

main()
{
   int i;
   int x = 0;
   int y = 0;
   opengraph();
   loadfont();
   h2 = open("normal.chr",O_CREAT|O_APPEND|O_BINARY);
    for (i = 0; i < 256 ; i ++) {
       PutBitMap(x,y,i,1);
       if (x > 650){
          x = 0;
  y += 21;
       }else{
  x += 9;
       }
    }
    x = 0;
    y = 0;
    for (i = 0; i < 256 ; i ++) {
       savepattern(x,y);
       if (x > 650){
          x = 0;
  y += 21;
       }else{
  x += 9;
       }
    }
    outtextxy(100,300,"Prese any key to Exit");
    getch();
    close(h2);
    closegraph();
```



รูปที่ 3
จอภาพขณะรันโปรแกรม MAKEPAT

NORMAL.CHR มีขนาดข้อมูลทั้งหมด 12288 ไบต์ เรานำ 256 ไปหาร จะได้ขนาดของอะเรย์ที่ใช้เก็บตัวอักษร 1 ตัว ซึ่งมีค่าเท่ากับ 48 ไบต์พอดี สำหรับแฟ้มข้อมูลชื่อ NORMALL.CHR ผู้เขียนไม่ใส่ในซอฟต์เมล์ เพื่อให้ผู้อ่านได้ลองรันโปรแกรมด้วยตัวเอง

โปรแกรมถัดไป เป็นตัวอย่างการนำตัวอักษรมาแสดงบนจอภาพ (ลิสติงที่ 3)

จากลิสติงที่ 3 เป็นตัวอย่างง่าย ๆ ที่จะนำตัวอักษรที่ได้มาแสดงบนจอภาพเป็นแนวทางให้ผู้อ่านนำไปประยุกต์ใช้งานต่อไป ข้อสังเกตคือ คำสั่ง showchar ในลิสติงที่ 3 จะต่างจากในลิสติงที่ 1 มาก เพราะจะเห็นว่าในลิสติงที่ 3 ใช้เพียงคำสั่ง putimage เพียงคำสั่งเดียวก็สามารถแสดงตัวอักษรได้ เหตุที่ผู้เขียนไม่ใช้คำสั่ง putimage เลยทันที ก็เพราะต้องการให้เห็นภาพพจน์การทำงานในส่วนนี้ชัดเจนขึ้น

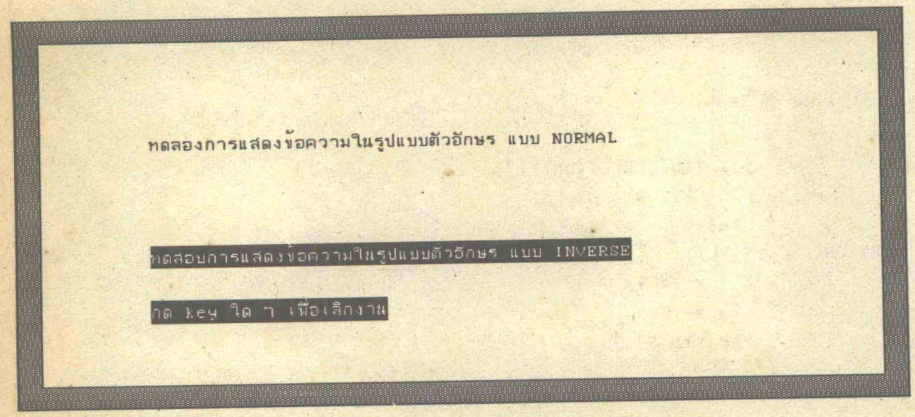
ในฟังก์ชัน outthaixy จะใช้แทนคำสั่ง outtextxy ของบอร์แลนด์เพื่อใช้

ลิสติงที่ 3
โปรแกรม TEST.C

```
#include <graphics.h>
#define  BUFFER   12288
#define  NORMAL   0
#define  INVERSE  1
unsigned char font[256][48];

void showchar(int x, int y,unsigned char ch,int OPTION)
{
  putimage(x,y,font[ch],OPTION);
}

void outthaixy(int x,int y,unsigned char *content,char attibute)
#define THAICHECK(s) ((s==209)||((s>=212)&&(s<=219))||((s>=231)&&(s<=237)))
{
   int OPTION;
   register int k,l;
   unsigned char ch;
   l = strlen(content);
   for(k = 0;k < l;k++){
      if(((x+9) > getmaxx())||((y+20) > getmaxy())){
         break;
      }
      ch = content[k];
      switch(attibute){
            case INVERSE:
                 {
                   if (THAICHECK(ch)) {
      OPTION = XOR_PUT;
                   }else{
                       OPTION = NOT_PUT;
                       x += 9;
                   }
                   break;
                 }
            case NORMAL:
                 {
                    if (THAICHECK(ch)){
                       OPTION = OR_PUT;
                    }else{
      OPTION = COPY_PUT;
                       x += 9;
                    }
                    break;
                 }
      }
```

ลิสติงที่ 3 (ต่อ)

```
        showchar(x,y,ch,OPTION);

    }
}

void opengraph(int driver)
{
    int mode;
    initgraph(&driver,&mode,"");
}

main()
{
    int f1;
    char test1[] = "ทดลองการแสดงข้อความในรูปแบบตัวอักษร แบบ NORMAL";
    char test2[] = "ทดสอบการแสดงข้อความในรูปแบบตัวอักษร แบบ INVERSE";
    opengraph(0);
    if (graphresult() != -1) {
if ((f1 = open("normal.chr",1)) != -1) {     /* 1 = O_RONLY */
        read(f1,font,BUFFER);
        close(f1);
      }
      rectangle(20,20,getmaxx()-20,getmaxy()-20);
      setfillstyle(9,YELLOW);
rectangle(0,0,getmaxx(),getmaxy());
floodfill(1,1,YELLOW);
      outthaixy(100,100,test1,NORMAL);
      outthaixy(100,200,test2,INVERSE);
      outthaixy(100,250,"กด key ใด ๆ เพื่อเลิกงาน",INVERSE);
      getch();
    }
    closegraph();
}
/***************/
```



รูปที่ 4 จอภาพจากการรันโปรแกรม TEST

แสดงภาษาไทย ในฟังก์ชันนี้จะมีคำสั่ง

```
# define    THAICHECK(s)
((s==209)::((s>=212)&&(s<=219)
::((s>=231)&&(s<=237)))
```

คำสั่งนี้จะทำหน้าที่ตรวจสอบว่าอักขระที่จะแสดงบนจอภาพเป็นตัวอักษรล่าง หรือตัวอักษรบนหรือไม่ ฟังก์ชันนี้จะแสดงตัวอักษรได้ 2 ลักษณะคือ แบบตัวปกติ (NORMAL) และ ตัวขาวกลับดำ (INVERSE) ผลที่ได้ของโปรแกรมนี้จะมีลักษณะดังรูปที่ 4

ก่อนจาก

สำหรับครั้งต่อไป จะเป็นรายละเอียดฟังก์ชันต่าง ๆ ที่มีใช้ในระบบไร่เกษตรธรรม ไทย กราฟิก ไลบรารี โปรแกรมที่ปรากฏในบทความนี้ ผู้เขียนได้บรรจุในซอฟต์เมล์แล้ว ขอให้ติดต่อกับทางกองบรรณาธิการครับ